

## JOSSE Lessons Learned Chapter 1

Philip Johnson  
Collaborative Software Development Laboratory  
Information and Computer Sciences  
University of Hawaii  
Honolulu HI 96822

(1)

## Plan for (at least) two versions

The first design/implementation cycle of a sophisticated system is where you learn what the real design should be.

You must have the courage, patience, and persistence to redesign and reimplement as you learn more if you want to satisfy the three prime directives.

Every version I reviewed could (and should) be improved.

I did not catch every error in every assignment.

(2)

## Types of problems

### JavaDoc

- Jbuilder template
- Uninformative summary lines
- Uninformative application/package overview
- "Description of" JRefractory comments
- Incomplete sentences

### Coding/build standards

- Capitalization issues
- Checkstyle/Javadoc build failures
- Detritus (.local, ~, build/, etc.) in zip dist.
- Lack of self-review

(3)

## Types of problems (cont.)

### Design

- Poor container data structure choice
- Poor design of enumerated types
- Use of both subclassing and enum type
- Lack of information hiding
- Needless complexity
- Inappropriate access control
- Inappropriate use of RuntimeException
- Default constructor creates invalid object
- Printing to system.out in normal operation
- Coupling of data to container

(4)

## Jbuilder doc templates are bogus

Whoever designed them doesn't grok javadoc.



(5)

## Uninformative summary lines

"Account class" not a good javadoc class summary line.

Guidelines for good summary lines:

- EJS Chapter 4
- jdk1.3 java.util classes.

(6)

## Uninformative application and/or package overview

The application summary (overview.html) should explain:

- the purpose of the application.
- how the interior package(s) work together to accomplish the requirements.

The package summary should explain:

- the purpose of the class
- how the interior class(es) work together to accomplish the package's purpose.

(7)

## "Description of" from JRefactory

These are not helpful to the reader.



(8)

## Incomplete sentences

Make sure your writing is:

- forceful, clear, concise.

However, use complete sentences.

(9)

## Capitalization Issues

Things that should be capitalized:

- First word of sentences.
- Class names

Things that shouldn't be capitalized:

- Method names

(10)

## Checkstyle/JavaDoc failures

Before turn-in:

- Run 'ant checkstyle'
- Run 'ant javadoc'

Read the output and make sure that there are no errors reported by checkstyle or javadoc (regardless of whether or not build is "successful".)

(11)

## Detritus in zip

Distributions should not include:

- backup (\*~) files
- build directory and its subdirectories.
- \*.jpx.local files

After making a distribution, try doing:

- jar -tf <dist>.zip

Check to make sure you have exactly what is needed and appropriate.

(12)

## Lack of self-review

To find many errors, do the following:

- Create javadocs and actually read them.
- Create j2h and actually look at it.

You'll be amazed at what you can improve by actually reading your own work.

(13)

## Poor container data structure choice

Sequential search is:

- inefficient at run-time
  - requires more lines of code to implement.
- ArrayList is thus a bad data structure choice.

Better: one of the Map data structures.

References:

- jdk1.3 java.util JavaDocs
- Java in a Nutshell, Chapter 23
- ICS 311

(14)

## Poor design of enumerated types

Instead of:

- A string that takes the values "Faculty", "Student", "Administrator"
- An int that takes the values 0, 1, 2

You should use the typesafe enum pattern:

- <http://developer.java.sun.com/developer/Books/effectivejava/Chapter5.pdf>

(15)

## Use of both subclassing and enumerated type

Some programs had both:

- account superclass, with subclasses Faculty, Student, Administrator
- An enumerated type representing Faculty, Student, Administrator

This is an example of a "redundant representation", which leads to:

- maintenance problems.
- understandability problems.

Instead, use one representation only.

(16)

## Lack of information hiding

Your container class should hide the internal representation chosen to store the instances in.



(17)

## Needless complexity

Some solutions were "excessively factored" and difficult to understand.

This happens to everyone occasionally.

After you understand the problem, see if you can simplify your solution.

(18)

## Inappropriate access control

Think about proper use of public, private, package-private, and protected.

- Public: available to every client in every package—part of the external API of the system.
- Private: an internal feature of the class
- Package-private: available only within the package.
- Protected: available only to subclasses.

(19)

## Inappropriate use of RuntimeException

RuntimeException should be used only for unexpected errors indicating a flaw in program logic (EJS, Rule 83).

An invalid password is not one of these situations.

AccountInfoException should be a checked, not an unchecked, exception.

(20)

## Default constructor creates invalid object

In this application, you shouldn't provide a default (no arg) constructor for Faculty, Student, Administrator classes since it can't construct a valid object.

(21)

## Printing to System.out in normal operation

Your system should not print to System.out under "normal operating conditions".

Exceptions:

- Catastrophic failure
- Test code
- Debugging output

You should use logging facilities to provide an audit trail.

(22)

## Coupling of data to container

Some systems tightly coupled the account data structures to the container data structures, usually via something like this:

- AccountInfo.add(Name, email, password, type)
  - The account instance was created internally.

It is better to decouple them.

- The Account representations "stand on their own" and can be used without a container.
- Multiple kinds of containers can be implemented to hold account objects.

(23)

## Assignment 2

Task:

- Work in groups of two
- Use CVS and JUnit
- Start with one of your implementations, and make it better.
- Smallness, cleanness, simplicity are all virtues!

(24)